Citation for published version:
Kelly, DM, Chen, Q & Zang, J 2015, 'PICIN: a particle-in-cell solver for incompressible free surface flows with two-way fluid-solid coupling', *SIAM Journal on Scientific Computing*, vol. 37, no. 3, pp. B403-B424. https://doi.org/10.1137/140976911

DOI:
10.1137/140976911

Publication date:
2015

Link to publication

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. May. 2023

SIAM J. SCI. COMPUT.
Vol. 37, No. 3, pp. B403–B424
© 2015 Society for Industrial and Applied Mathematics

PICIN: A PARTICLE-IN-CELL SOLVER FOR INCOMPRESSIBLE FREE SURFACE FLOWS WITH TWO-WAY FLUID-SOLID COUPLING*

D. M. KELLY†, Q. CHEN‡, AND J. ZANG§

Abstract. This paper details a novel numerical approach for solution of the Navier–Stokes equations for free surface flows involving two-way fluid-solid interaction in arbitrary domains. The approach, which is hybrid Eulerian Lagrangian in nature, is based on the full particle particle-in-cell (PIC) method applied to incompressible flows. An extension of the distributed Lagrange multiplier (DLM) technique proposed by Patankar et al. [*Int. J. Multiphase Flow*, 26 (2000), pp. 1509–1524] is employed for the two-way fluid-solid coupling. The resulting code is called PICIN. Solid bodies can be mobile, either having prescribed motion or moving as a consequence of interaction with the fluid. Numerical results for three distinct example applications of the model in two spatial dimensions are given. A comparison of PICIN predictions with state-of-the-art numerical results of other researchers is made for each of the test cases presented.

Key words. computational fluid dynamics, Navier–Stokes, particle-in-cell, SPH, VOF, level set, incompressible fluid, fluid-structure interaction

AMS subject classifications. 35Q30, 65M06

DOI. 10.1137/140976911

1. Introduction.

The increase in affordable computing power that has occurred over the last decade has made possible the numerical solution of the Navier–Stokes equations in both two and three spatial dimensions for a wide variety of real-world free-surface flows. While pure Eulerian methods have been used to solve the Navier–Stokes equations (see, e.g., [18, 16]), a Lagrangian approach is better suited to complex free surface flows as it handles both wave breaking and wetting drying processes naturally. Pure Lagrangian methods such as the moving-particle semi-implicit (MPS) and smoothed particle hydrodynamic (SPH) schemes have become increasingly popular (see, e.g., [17, 23]). Of these, SPH has become the Lagrangian method of choice and has been used to simulate complex free surface flows with considerable success; see, e.g., [21]. Both weakly compressible SPH (WCSPH) and incompressible SPH (ISPH) are computationally extremely demanding in terms of CPU time [15]. The work presented in this paper was motivated by developing a code for industrial application, the idea being that the code should have all the flexibility of SPH with the efficiency of an Eulerian approach.

*Submitted to the journal's Computational Methods in Science and Engineering section July 10, 2014; accepted for publication (in revised form) March 31, 2015; published electronically June 2, 2015.

http://www.siam.org/journals/sisc/37-3/97691.html

†Senior Engineer, HR Wallingford, Wallingford, Oxon, OX10 8BA, UK. Current address: Asst. Prof. of Research, International Hurricane Research Center, Florida International University, Miami, FL 33199 (dakelly@fiu.edu). This author's work was supported by HR Wallingford.

‡Research Unit for Water, Environment and Infrastructure Resilience (WEIR), Department of Architecture and Civil Engineering, University of Bath, Bath BA2 7AY, UK (qc296@bath.ac.uk). This author's work was supported by the University of Bath and HR Wallingford for his Ph.D. study of which this work forms a part.

§Director of the Research Unit for Water, Environment and Infrastructure Resilience (WEIR), Department of Architecture and Civil Engineering, University of Bath, Bath BA2 7AY, UK (jz235@bath.ac.uk). This author's work was supported by the University of Bath and HR Wallingford.

The particle-in-cell (PIC) approach was invented at the Los Alamos National Laboratory in 1955 by Francis Harlow [12] and was further developed there until it became the practical methodology described in [13]. The PIC method was originally intended for compressible fluid flows, and [14] developed the marker and cell (MAC) method for incompressible fluids. The classical PIC method described in [13] suffers from severe numerical diffusion. This is due to the two transfers of the velocity field from the particles to the grid and back to the particles at every time-step. To reduce this numerical diffusion, [5] suggested incrementing the particle velocity field based on the change in the grid velocity field once the Eulerian computation has been conducted. This approach necessitates that the velocity field be stored on the grid at the end of each Eulerian computation to be used at the next time-step. In practice the Eulerian grid can be relatively coarse, so memory requirements for storage are not excessive. Particle velocities are incremented by interpolation of the change in the fixed-grid velocity field to the particle locations. The real novelty of the (full particle) PIC method is the fact that the particles are used to track the free surface *and* advect the velocity field. This both is computationally efficient and leads to far less artificial diffusion than an Eulerian advection step. The first published application of (full particle) PIC to incompressible free surface flow is due to [35]. The full particle PIC algorithm described here is based on that given in [35] and uses the fractional step method for time integration, employing the classic projection method of Chorin [8]. The projection technique enforces incompressibility and allows the velocity and pressure fields to be computed distinctly. The time-step in full particle PIC is based solely on the fluid velocity field, and the speed of sound plays no part in determining the stability criteria in the PICIN code. Moreover, as the pressure Poisson equation (PPE) is solved on an underlying simple Cartesian Eulerian mesh, its solution can be optimized.

In PICIN, two-way fluid-structure interaction is handled via the distributed Lagrange multiplier (DLM) approach suggested by [30]. This approach is both simple to implement and computationally efficient. The basis of the approach is to solve the entire fluid-solid domain as if it were a uniform incompressible fluid. Correction for the density difference between the fluid and solid is then made. Finally, the deformation rate tensor within solid regions is constrained to zero, thus forcing rigid body motion in solid regions. This is achieved by using the average velocity in these regions, which provides a unique solid velocity which eliminates the translational $\boldsymbol{U}$ and angular velocity $\boldsymbol{\omega}$ of the solid from the governing equations. The approach requires modification when implemented within the hybrid Eulerian–Lagrangian framework used by PICIN.

The paper is organized as follows: section 2 gives the equations that govern the coupled fluid solid motion. Section 3 details the numerical method of solution, including details of how the free surface boundary, domain boundaries, and two-way fluid-solid coupling are handled within the hybrid Eulerian–Lagrangian framework used by PICIN. In section 4 the results of three distinct test cases are presented and discussed. These test cases have been chosen to illustrate the flexibility of the PICIN approach. Finally, conclusions are drawn in section 5.

2. Governing equations. PICIN employs the strong form of the governing equations for two-way coupled fluid-solid motion proposed by [30]. Within this framework, the computational domain is considered to contain both the fluid and any solid bodies and is denoted by Ω. The fluid and solid domains are subsets of Ω and are denoted by $\mathbb{F}$ and $\mathbb{S}$, respectively. On the boundary of Ω, denoted by Γ, problem-

specific boundary conditions are enforced. Refer to Figure 1 for an overview of the solution domain. In vector notation, the system of equations governing the fluid and solid motion has the following form:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \boldsymbol{f} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} \quad \text{in } \mathbb{F}, \tag{2.2}$$

and

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \boldsymbol{f} - \frac{1}{\rho_S}\nabla p + \nabla \cdot \Pi \quad \text{in } \mathbb{S}, \tag{2.3}$$

with the boundary conditions

$$\boldsymbol{u} = \boldsymbol{u}_\Gamma(\boldsymbol{x}, t) \quad \text{on } \Gamma(\boldsymbol{x}, t) \tag{2.4}$$

and

$$\boldsymbol{u} = \boldsymbol{u}_i \text{ and } p = 0 \quad \text{on } \zeta(\boldsymbol{x}, t), \tag{2.5}$$

where $\zeta = \zeta(\boldsymbol{x}, t)$ is the free surface and

$$\boldsymbol{u} = \boldsymbol{u}_i \text{ and } \boldsymbol{\Pi} \cdot \hat{n} = \boldsymbol{T} \quad \text{on } \partial\mathbb{S}(t), \tag{2.6}$$

which implies a no-slip condition on the boundary of $\mathbb{S}$, here denoted as $\partial\mathbb{S}$. We note that when using the DLM approach to handle solids it is not necessary to enforce (2.6) explicitly; instead, this boundary condition is implicitly satisfied [30]. In this paper we consider two spatial dimensions only for which $\boldsymbol{u} = [u, w]^T$ is the velocity field (u and w being the horizontal and vertical components, respectively), p is the pressure, $\boldsymbol{f} = [g_x, g_z]^T$ represents the vector of body forces acting on the water due to gravity, ρ is the water density, ρ_S is the solid density, ν is the kinematic viscosity of water, and $\boldsymbol{\Pi}$ is the stress tensor. The traction force of the fluid on the solid, denoted by $\boldsymbol{T}$, is the sum of the projected viscous stresses and pressure. In the absence of any solid deformation, $\boldsymbol{\Pi}$ can be written as $\boldsymbol{\Pi} = -p\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity tensor.

We also consider a fluid signed distance function (SDF) ϕ, which is used to identify fluid regions and define the free surface boundary. The SDF is governed by a simple level set function [33].

The value of the SDF ϕ is set such that $\phi < 0$ in water, $\phi > 0$ in air, and the iso-contour $\phi = 0$ denotes the free surface ζ. The same approach is employed for the solid SDF ϕ_S, which is used to identify solid regions $\phi_S < 0$ and the solid boundaries for which $\phi_S = 0$.

While the level set function can be solved directly in an Eulerian sense, the PICIN solver treats the evolution of the fluid and solid SDFs in a Lagrangian manner using the fluid or solid particles to find the zero iso-contour for ϕ and ϕ_S (and thus evolve the level set). We note that, like the pressure, the fluid SDF is computed at cell centers. The solid SDF is computed at cell vertices (see Appendix A).

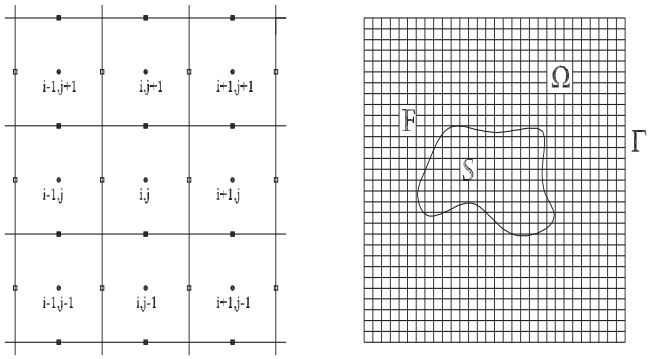

FIG. 1. *Computational grid illustrating cell centers (circles) where pressures, p, and signed distances, ϕ, are computed and vertical (white squares) and horizontal faces (black squares) where the x and z velocity components are computed respectively (left). Computational domain Ω with the domain boundary Γ and embedded solid $\mathbb{S}$ and fluid $\mathbb{F}$ regions (right).*

3. Full particle PIC numerical method for incompressible flow.

3.1. The PICIN algorithm. The Navier–Stokes equations for the fluid (2.2) are solved, subject to the incompressibility constraint (2.1), everywhere in Ω. Solution in PICIN is via the time-operator-splitting (pressure projection) technique [8, 35]. Chorin [8] first proposed the use of time-operator-splitting to the full Navier–Stokes equations in order to decouple $\boldsymbol{u}$ and p. This approach separates the convection-diffusion part of the Navier–Stokes equations from the incompressibility part. The PICIN algorithm comprises four Eulerian steps followed by a Lagrangian step that both advects the velocity field and tracks the free surface boundary. In PICIN the purpose of the underlying Eulerian grid is to increase efficiency when accounting for the body forces, applying the domain and free surface boundary conditions and ensuring that the fluid remains incompressible. The first stage in the grid calculation is to interpolate the velocity field from the particles onto the grid. Following [14], a staggered grid is used, where the pressure is stored at the center of the grid cells and the u and w components of the velocity are stored on the vertical and horizontal vertices, respectively (see Figure 1). Such a grid ensures that second-order accurate central differencing in space can be employed without introducing false local maxima or minima.

The first Eulerian step is straightforward. Apply body forces:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{f}. \tag{3.1}$$

Equation (3.1) is integrated in time using the first-order accurate explicit Euler method, i.e.,

$$\boldsymbol{u}^1 = \boldsymbol{u}^n + \Delta t \boldsymbol{f}. \tag{3.2}$$

The next step is to apply the velocity diffusion due to the fluid viscosity to compute the tentative velocity field $\tilde{\boldsymbol{u}}$:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \nu \nabla^2 \boldsymbol{u}^1. \tag{3.3}$$

Currently a simple explicit forward-in-time centered-in-space (FTCS) difference scheme is employed to integrate (3.3). Thus, diffusion of the x-component of velocity at point $i+\frac{1}{2}, j$ on the mesh is computed as

$$\tilde{u}_{i+\frac{1}{2},j} = u^1_{i+\frac{1}{2},j} + \nu\Delta t\left(\frac{u^1_{i+\frac{3}{2},j} + u^1_{i-\frac{1}{2},j} - 2u^1_{i+\frac{1}{2},j}}{(\Delta x)^2} + \frac{u^1_{i+\frac{1}{2},j+1} + u^1_{i+\frac{1}{2},j-1} - 2u^1_{i+\frac{1}{2},j}}{(\Delta z)^2}\right). \tag{3.4}$$

Diffusion of the z-component of velocity is approximated in a similar manner. An implicit scheme has also been coded in PICIN to solve (3.3) via Gauss–Seidel iteration with successive overrelaxation (SOR). Numerical experiments have shown that, when using relatively small time-steps, the explicit scheme is satisfactory for the vast majority of problem types, and it is used here. The tentative velocity field, $\tilde{\boldsymbol{u}}$, is unlikely to be divergence-free; it is therefore necessary to compute the pressure. In the pressure projection approach the pressure is a Lagrange multiplier used to ensure that $\check{\boldsymbol{u}}$ satisfies (2.1). The PPE is constructed following [9]. Once the body forces and diffusion have been integrated, the remaining term from (2.2) is

$$\frac{(\check{\boldsymbol{u}} - \tilde{\boldsymbol{u}})}{\Delta t} = -\rho^{-1}\nabla p; \tag{3.5}$$

as ρ and Δt are constant, it is possible to introduce $\varphi = -\Delta t\rho^{-1}p$ to give

$$\check{\boldsymbol{u}} = \tilde{\boldsymbol{u}} + \nabla\varphi. \tag{3.6}$$

Taking the divergence of both sides of (3.6) and recalling that we require that the divergence of the new velocity field, $\check{\boldsymbol{u}}$, be zero gives a Poisson equation for φ:

$$\nabla^2\varphi = \nabla\cdot\tilde{\boldsymbol{u}}. \tag{3.7}$$

Discretization of this form of the PPE is effected using a finite difference approach that incorporates the treatment of the free surface and any domain boundaries. We note that, due to the modification to the PPE described in section 3.2, the domain boundaries $\Gamma = \Gamma(\boldsymbol{x}, t)$ do not have to be grid-aligned. We note also that if there is no two-way fluid-solid coupling, then $\boldsymbol{u}^{n+1} = \check{\boldsymbol{u}}$. If fluid-solid coupling is present in the flow, then additional steps are required to obtain $\boldsymbol{u}^{n+1}$; these are detailed in section 3.2.

3.2. Boundary conditions and solution of the PPE. For a projection-based solution of the Navier–Stokes equations it is sensible to impose boundary conditions during the pressure solve. In PICIN a signed distance field is utilized to ensure the correct treatment of both the free surface and any domain boundaries that are not treated using the grid-aligned approach. The various approaches used in PICIN to treat different boundary scenarios are detailed below.

Free surface boundary. For the free surface boundary a Dirichlet-type zero pressure boundary condition is imposed. Tracking of the free surface is performed using the particles. Following [35] the free surface boundary is identified by employing a particle-based signed distance function (SDF) $\phi = \phi(\boldsymbol{x}, t)$ constructed using a fast-sweeping approach [34]. The SDF with respect to the free surface is computed at

cell centers. To avoid nonphysical artifacts at the air-water interface it is important to apply the zero pressure boundary condition at the actual free surface as opposed to the center of a surface cell [27]. In PICIN, the zero pressure condition is applied directly at the free surface through use of ϕ which is employed to modify coefficients on the left-hand side of (3.7). Using the method of [11], a second-order accurate scheme is adopted by defining ghost values of air pressure to apply a Dirichlet boundary condition on the actual free surface. For completeness we detail the approach here. Consider, without loss of generality, the solution of a one-dimensional (1D) PPE, $p_{,x,x} = \tilde{u}_{,x}$, where subscripts denote partial derivatives. The free surface interface is located at x_f with $x_i < x_f < x_{i+1}$. To facilitate explanation we define a coefficient $\Theta = \frac{(x_f - x_i)}{\Delta x}$. Since the interface x_f will be cut by the stencil in a general case, the standard second-order accurate PPE discretization should be remedied by defining

$$p_{i+1} = \frac{(\Theta - 1)}{\Theta} p_i. \tag{3.8}$$

Note that, in practice, in the PICIN code, we modify the denominator of (3.8) to $(\Theta + \epsilon\Delta x)$, where ϵ is a user-defined parameter included so that (3.8) remains bounded (for the test cases presented in this paper we use $\epsilon = 0.001$). This leads to a symmetric discretization of

$$\frac{(\frac{0-p_i}{\Theta\Delta x}) - (\frac{p_i - p_{i-1}}{\Delta x})}{\Delta x} = (\tilde{\boldsymbol{u}}_{,x})_i. \tag{3.9}$$

This equation enables the use of a standard PPE solver by simply modifying the coefficient in the left-hand side of the PPE. All results presented here used the biconjugate gradient method [31] to solve the PPE.

Grid-aligned solid boundaries. For the staggered Cartesian mesh described above, free-slip conditions are simple to implement at solid boundaries that are aligned with the underlying grid. The grid-aligned boundary in PICIN adopts hybrid Neumann and Dirichlet conditions. A zero normal velocity is prescribed on the boundary cell edges, and the pressure gradient condition is enforced in a way that only the coefficient in the PPE is changed. For ease of explanation we use a 1D discretization of the PPE as a basis to explain the method. In one dimension the discrete form of the PPE is written as

$$\frac{\Delta t}{\rho} \frac{p_{i+1} + p_{i-1} - 2p_i}{(\Delta x)^2} = \frac{\tilde{u}_{i+\frac{1}{2}} - \tilde{u}_{i-\frac{1}{2}}}{\Delta x}. \tag{3.10}$$

Assuming that $u_{i+\frac{1}{2}}$ is on the boundary cell edge and $\Gamma = \Gamma(\boldsymbol{x})$ only, then the boundary velocity should be zero at the next, $n+1$, time-step. Thus, we have

$$\check{u}_{i+\frac{1}{2}} = \tilde{u}_{i+\frac{1}{2}} - \frac{\Delta t}{\rho} \frac{p_{i+1} - p_i}{\Delta x} = 0. \tag{3.11}$$

Applying this to (3.10) yields

$$\frac{\Delta t}{\rho} \frac{p_{i-1} - p_i}{(\Delta x)^2} = \frac{-\tilde{u}_{i-\frac{1}{2}}}{\Delta x}. \tag{3.12}$$

Thus, in PICIN, only the coefficients in the PPE are altered to account for the existence of a grid-aligned solid boundary.

The tentative velocity field $\tilde{\boldsymbol{u}}$ is then projected onto the nearest divergence-free velocity field $\check{\boldsymbol{u}}$ using the pressure p as a Lagrange multiplier, via $\varphi = -\Delta t \rho^{-1} p$, to give

$$\check{\boldsymbol{u}} = \tilde{\boldsymbol{u}} + \nabla\varphi. \tag{3.13}$$

Non–grid-aligned solid boundaries. In order to treat stationary solid boundaries that are not grid-aligned, the divergence theorem is applied to both sides of the integral form of the PPE. This leads to the following rewriting of the PPE:

$$\oint \nabla\varphi \cdot \hat{\boldsymbol{n}}\, dA = \oint \tilde{\boldsymbol{u}} \cdot \hat{\boldsymbol{n}}\, dS, \tag{3.14}$$

where dA is the area differential of the cut-cell (cell containing a fraction of the solid boundary) and dS is the length differential. The right-hand side of (3.14) is approximated as

$$\begin{aligned}\oint \nabla\varphi \cdot \hat{\boldsymbol{n}}\, dA \approx S_{i-\frac{1}{2},j} \cdot \frac{(\varphi_{i,j} - \varphi_{i-1,j})}{\Delta x} + S_{i+\frac{1}{2},j} \cdot \frac{(\varphi_{i,j} - \varphi_{i+1,j})}{\Delta x} \\ + S_{i,j-\frac{1}{2}} \cdot \frac{(\varphi_{i,j} - \varphi_{i,j-1})}{\Delta y} + S_{i,j+\frac{1}{2}} \cdot \frac{(\varphi_{i,j} - \varphi_{i,j+1})}{\Delta y},\end{aligned} \tag{3.15}$$

and the left-hand side is approximated as

$$\oint \tilde{\boldsymbol{u}} \cdot \hat{\boldsymbol{n}}\, dS \approx S_{i-\frac{1}{2},j} \cdot \tilde{\boldsymbol{u}}_{i-\frac{1}{2},j} - S_{i+\frac{1}{2},j} \cdot \tilde{\boldsymbol{u}}_{i+\frac{1}{2},j} + S_{i,j-\frac{1}{2}} \cdot \tilde{\boldsymbol{u}}_{i,j-\frac{1}{2}} - S_{i,j+\frac{1}{2}} \cdot \tilde{\boldsymbol{u}}_{i,j+\frac{1}{2}}, \tag{3.16}$$

where $S_{i\pm\frac{1}{2},j}$ and $S_{i,j\pm\frac{1}{2}}$ are the length fractions of each cell side that are open to flow. This procedure is simply an application of the finite volume approach first suggested by [28] to both sides of the PPE. The approach was formally investigated in terms of order of convergence by [26]. Computation of the length fractions open to flow S uses the solid SDF ϕ_S. Details of this procedure are given in Appendix A.

Advection. Once the divergence-free velocity field $\check{\boldsymbol{u}}$ has been obtained in Ω using (3.13), it is used to increment the particle velocities $\boldsymbol{u}_{gp}$. The value of $\boldsymbol{u}_{gp}$ is incremented using the approach outlined in section 3.4. When there is no two-way fluid-solid coupling all the particles in Ω are advected in a Lagrangian manner according to

$$\frac{D\boldsymbol{x}_p}{Dt} = \boldsymbol{u}_{gp}, \tag{3.17}$$

using a third-order accurate Runge–Kutta method for the time integration [32]. We note that over time truncation errors can lead to particle distribution errors. For this reason we regularize the particle distribution after a given number of time-steps. Full details of this procedure are given in section 3.5.

Two-way fluid-solid coupling. If there are solid bodies present in Ω, then additional computations, comprising a tentative advection step and a density correction step, must be performed before the main advection step (3.17) is executed. Tentative advection is only carried out on particles that are inside the solid and in the domain immediately surrounding the solid; this is shown schematically in Figure 2. These

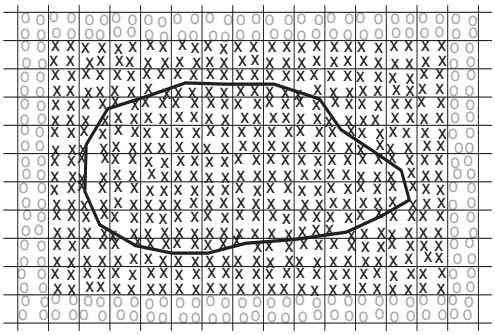

FIG. 2. *Schematic close-up of mesh and particles around a solid body. The × particles are used in the tentative advection step to compute $\check{\boldsymbol{u}}$; the o particles are not.*

particles are used to advect the velocity field and transfer the advected velocity field to the grid. The particles are then moved back to their original positions. This tentative advection step is used to update $\check{\boldsymbol{u}}$. The next step is to account for the effect of relative density and any momentum transfer due to solid-solid interaction in $\mathbb{S}$. This is achieved following [30] by correcting $\check{\boldsymbol{u}}$ so it accounts for these effects according to

$$\hat{\boldsymbol{u}} = \check{\boldsymbol{u}} + \theta \frac{\Delta t}{\rho_S} \boldsymbol{S}, \tag{3.18}$$

where the source term $\boldsymbol{S}$ is given by

$$\boldsymbol{S} = \rho_S \boldsymbol{A}_c + \boldsymbol{r}_i \times \rho_S \boldsymbol{\alpha_c} - (\rho_S - \rho)\left\{ \frac{D\check{\boldsymbol{u}}}{Dt} - \boldsymbol{g} \right\}; \tag{3.19}$$

here $\boldsymbol{A}_c$ and $\boldsymbol{\alpha_c}$ are the translational and angular accelerations, respectively, due to any solid-solid collisions that occur. These accelerations are computed using a summation of all the forces acting on that body over a time-step divided through by the mass of the solid body. The vector $\boldsymbol{r}_i$ is the position vector of the point with respect to the center of mass of the solid. Cells that are only partially filled by the solid body are handled using a solid volume weighting term $0 \leq \theta \leq 1$. This term defines the fraction of a cell that is occupied by a solid. Calculation of the solid volume weighting term is somewhat involved, and the procedure used to determine θ in PICIN is given in Appendix B.

The material derivative in (3.19) is currently approximated in a purely Eulerian manner.

In PICIN we rewrite the x- and z-components of the advection term in the material derivative, using the chain rule and (2.1), as

$$u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = \frac{\partial (u^2)}{\partial x} + \frac{\partial (uw)}{\partial z} \tag{3.20}$$

and

$$u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} = \frac{\partial (uw)}{\partial x} + \frac{\partial (w^2)}{\partial z}, \tag{3.21}$$

which gives us the advection terms in conservation form. We solve this form of the advection term using first-order accurate Euler integration in time and second-order accurate centered differencing in space.

Once (3.18) has been evaluated in $\mathbb{S}$, the total momentum in the solid region is correct. A force is required to correct the velocities in $\mathbb{S}$ so that they maintain the rigidity of the solid. Patankar [29] noted that because the momentum in the rigid body must be conserved, the unique constant velocity in $\mathbb{S}$ is simply the average of the translational velocities computed using

$$M\hat{\boldsymbol{U}} = \int_{\mathbb{S}} \rho_S \hat{\boldsymbol{u}}\, dx, \tag{3.22}$$

where M is the mass of the solid. The average of the angular velocities is also required. For a solid $\boldsymbol{r}$

$$I_p \hat{\boldsymbol{\omega}} = \int_{\mathbb{S}} \boldsymbol{r} \times \rho_S \hat{\boldsymbol{u}}\, dx, \tag{3.23}$$

where I_p is the moment of inertia of the solid. Following [29], the rigid velocity $\hat{\boldsymbol{u}}_R$ is simply

$$\hat{\boldsymbol{u}}_R = \hat{\boldsymbol{U}} + \boldsymbol{r} \times \hat{\boldsymbol{\omega}}. \tag{3.24}$$

Note that (3.22), (3.23), and (3.24) must be evaluated for each distinct solid region. The final velocity update in $\mathbb{S}$ is achieved via

$$\boldsymbol{u}^{n+1} = (1-\theta)\hat{\boldsymbol{u}} + \theta \hat{\boldsymbol{u}}_R. \tag{3.25}$$

Once $\boldsymbol{u}^{n+1}$ has been found using (3.25), the change in velocity on the grid is used to increment the particle velocities $\boldsymbol{u}_{gp}$. All the particles in Ω are advected in a Lagrangian manner according to (3.17). While the fluid particles are advected using a third-order Runge–Kutta method, the solid particles are advected using a first-order accurate Euler integration of (3.17). As the fluid particles have a velocity field defined on the grid (which varies in space for the given time-step) a high-order integration scheme is used to move these particles through the grid velocity field. For each solid, the solid particles all move with a single unique velocity at each time-step. Moving the solid using a time integration that is only first-order accurate is therefore appropriate. It is possible to use a high-order method that further couples the fluid-solid motion (such as a predictor-corrector scheme). Such an approach is computationally very costly, and our numerical experiments have shown that in the majority of cases it is unnecessary.

It is noted that all solid velocities are computed on the mesh at cell centers. The value of $\check{\boldsymbol{u}}$ is obtained at cell centers in a componentwise manner via linear interpolation from the relevant cell edges. The new solid velocities obtained using (3.25) are then transferred back to the cell sides in a consistent manner.

As the rigidity constraint is only applied within $\mathbb{S}$, the velocity field in the band of fluid cells immediately surrounding a solid will not necessarily be divergence-free. This is important as the particles will be advected in, and will advect, this velocity field. It is possible to re-solve the PPE in Ω after rigidity has been enforced in $\mathbb{S}$ and iterate until the solution converges. This approach is obviously computationally expensive. Another option is to use the divergence-free extrapolation described in section 3.3 to extrapolate velocities from the main body of the fluid to overwrite velocities in these cells. In practice we have found that it is not necessary to correct the velocities explicitly in these cells as they will be forced to be divergence-free at the next time-step.

3.3. Extrapolating the velocity field to air cells. Although the divergence-free velocity field is computed within the water (i.e., where $\phi < 0$), values of velocity may also be required in air cells adjacent to the free surface for the Lagrangian advection step. Note that the CFL condition described below (section 3.6) prohibits particles from moving more than one cell away from the free surface per time-step. In PICIN, velocity extrapolation is conducted around the free surface explicitly by an averaging procedure. Initially, all air cell velocities are zeroed. Next, all air cells which have at least one neighboring cell with a nonzero velocity are identified, and these cells are then assigned a velocity which is the average of their nearest cells with nonzero velocities. The averaging procedure takes advantage of the fluid SDF and is repeated to extend the velocities to an N cell band around the free surface. More details about this technique can be found in [1].

3.4. Transferring information to and from the grid. Finally, after they have been advected, velocities must be transferred from the particles to the Eulerian grid. Also, once the velocity field has been updated on the grid, $\frac{d\boldsymbol{u}}{dt}$ must be computed and interpolated to the particle positions in order to increment the particle velocities. To transfer velocities from the particles to the grid, a weighted sum of the particle velocity components is used:

$$\boldsymbol{u}_g = \sum\nolimits_p \sigma^{-1} \boldsymbol{u}_p S_n(\boldsymbol{x}_p - \boldsymbol{x}_g), \tag{3.26}$$

where S_n is an assignment function with bounded support and σ is a weighting function (see [5] for further details). The subscripts p and g represent values on particles and at the correct (in relation to the velocity component) cell edges, respectively. For this work a cubic spline assignment function [24] was used to transfer particle velocities to the grid. Particle velocity is incremented by interpolating the velocity change from the grid to the particle position using bilinear interpolation [31].

3.5. Particle redistribution. One problem of the PIC method is that the particle distribution in space tends to become uneven over time due to truncation error. This can ultimately lead to "holes" appearing in fluid cells, particles gathering together, and inaccurate velocity transfer. The problem is shared by other particle-based techniques such as SPH; see, e.g., [22]. To prevent this problem, following the idea of [2], an anisotropic particle resampling algorithm is used in PICIN. The particles are offset slightly in a manner that considers all surrounding particle positions. Thus, with h being the smoothing length associated with the assignment function, $\boldsymbol{x}_p$ is updated as

$$\Delta \boldsymbol{x}_{pi} = -\Delta t \gamma_s h \sum_j \frac{\boldsymbol{x}_{pj} - \boldsymbol{x}_{pi}}{\left\| \boldsymbol{x}_{pj} - \boldsymbol{x}_{pi} \right\|} S_n(\boldsymbol{x}_{pj} - \boldsymbol{x}_{pi}, h), \tag{3.27}$$

where γ_s is a constant stiffness coefficient. Simulation results are not unduly sensitive to the value of this coefficient; however, the coefficient should not be too large or too small as it may cause a large increase in CPU time or cause the model to fail to converge. In particular, fluid particles may cross solid wall boundaries if too large a value is used for γ_s. We note that the value of γ_s is problem-specific, and we use $\gamma_s = 10$ for all the simulations presented here.

The particle velocity is first transferred onto the grid and is then interpolated back after the particles are updated to their final new positions by

$$\boldsymbol{x}_{pnew} = \boldsymbol{x}_p + \Delta \boldsymbol{x}_p. \tag{3.28}$$

A carefully chosen criterion is required to terminate resampling. In PICIN, the average covariance of particles computed by

$$C_i = \frac{\sum_j (\boldsymbol{x}_{pj} - \bar{\boldsymbol{x}}_{pi})(\boldsymbol{x}_{pj} - \bar{\boldsymbol{x}}_{pi})^T S_n(\boldsymbol{x}_{pj} - \bar{\boldsymbol{x}}_{pi}, h)}{\sum_j S_n(\boldsymbol{x}_{pj} - \bar{\boldsymbol{x}}_{pi}, h)}, \tag{3.29}$$

where

$$\boldsymbol{x}_{pi} = \frac{\sum_j \boldsymbol{x}_{pj} S_n(\boldsymbol{x}_{pj} - \boldsymbol{x}_{pi}, h)}{\sum_j S_n(\boldsymbol{x}_{pj} - \boldsymbol{x}_{pi}, h)}, \tag{3.30}$$

is used. We stop resampling the particles when the sum of particle covariance converges to within a tolerance of 1×10^{-4}. We note that particles that belong to free surface interfaces are fixed while redistribution is undertaken. Similar approaches have also been employed in the SPH community; see, e.g., [19, 10]. However, these approaches shift the particle position based on the particle concentration gradient rather than the particle position gradient. It is noted that the resampling algorithm introduces additional numerical diffusion. Thus, in order to reduce this unwanted diffusion, the resampling algorithm is only applied every N_r time-steps. For the PICIN simulations presented in this paper, $N_r = 30$ unless stated otherwise.

3.6. Numerical stability. In the context of their MAC solver Markham and Proctor [20] suggested a simple stability criterion to determine Δt. In summary their scheme is to limit Δt such that any particle cannot traverse more than one cell boundary in either direction in one time-step. Thus, in the context of the full particle PIC method, we require that

$$\Delta t \leqslant \min\left(\frac{\Delta \boldsymbol{x}_{min}}{|\boldsymbol{u}_{max}|}\right). \tag{3.31}$$

The authors of [20] point out that determining $\boldsymbol{u}_{max}$ is not as trivial as it first seems. This is because the inequality $|\boldsymbol{u}^{n+1}_{max}| > |\boldsymbol{u}^{n}_{max}|$ could hold as a consequence of the effect of body forces. A conservative approach is to consider that

$$|\boldsymbol{u}_{max}| = |\boldsymbol{u}^{n}_{max}| + \Delta t|\boldsymbol{f}|. \tag{3.32}$$

Combined with the upper bound of (3.31), this gives

$$|\boldsymbol{u}_{max}| = |\boldsymbol{u}^{n}_{max}| + (\Delta \boldsymbol{x}_{min}|\boldsymbol{f}|)^{\frac{1}{2}}, \tag{3.33}$$

and thus

$$\Delta t \;=\; \min\left(\frac{\Delta \boldsymbol{x}_{min}}{\left(|\boldsymbol{u}^{n}_{max}| + (\Delta \boldsymbol{x}_{min}|\boldsymbol{f}|)^{\frac{1}{2}}\right)}\right) C_{FL}, \tag{3.34}$$

where $C_{FL} \leq 1$ is the Courant number. We thank one of the reviewers for pointing out that (3.34) may not be restrictive enough to ensure stability in all cases. Indeed, for problems involving fluid-solid interactions, the forces due to pressure gradients could lead to very large values of $\boldsymbol{u}^{n+1}_{max}$. Moreover, the time constraint due to viscous diffusion can become dominant for flows with high viscosity and/or flows that require a very fine mesh. Thus, it may be necessary to satisfy several time-step criteria to ensure model stability in all cases. The criteria given in [25] would appear appropriate.

4. Test cases. In this section simulation results for the PICIN solver are presented for three distinct free surface flow problems. The test cases illustrate the wide range of problems to which the PICIN solver can be applied. Moreover, the problems presented here require the implementation of a variety of different boundary conditions. The implementation of these boundary conditions within a hybrid Eulerian–Lagrangian framework is described. For each test case we provide the CPU time. Unless otherwise stated, results were obtained on an Intel Core$^{\text{TM}}$i7–4500U (CPU@1.80Ghz) laptop with 8Gb RAM employing a Windows OS. It is noted that the FORTRAN code used for this work is not optimized. We believe that the use of an optimized code will significantly reduce the CPU times presented here. A companion paper [7] will present comparisons of PICIN results with benchmark experimental data for a selection of two-dimensional (2D) flow problems.

Test 1: Dam break flow. Here we present results for the classical wet-dry dam break test as described in [15] and [21], the authors of which used the test to validate their respective SPH solvers. A column of water is contained within a tank, and the dam containing the water column is instantaneously released, resulting in the collapse of the water column due to gravity. The water flows across the horizontal bed and eventually impacts a vertical wall. The initial conditions are shown in Figure 3.

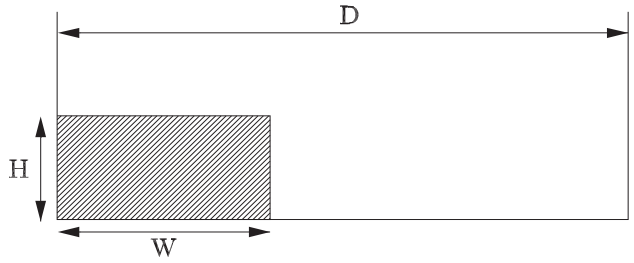

FIG. 3. *Problem set-up for Test* 1.

For the results presented here we used two different mesh resolutions, $\Delta x = \Delta z = H/40$ and $\Delta x = \Delta z = H/80$. Four particles were seeded randomly in each water cell. The Courant number was set to 0.5. The first four panels of Figure 4 show snapshots of the pressure field interpolated at particle locations for the same dimensionless times as those presented in Figure 3 of [21]. The final two snapshots show the results at later times following the collapse of the internal cavity. For this case $W = 2H$ and $D = 5.366H$. Similar snapshots of pressure are provided in [15], where the authors used both WCSPH and ISPH SPH models. The results of the δ-SPH model [21] are superior to those of the WCSPH and ISPH simulations presented in [15], which exhibit large amounts of nonphysical fragmentation.

It is clear from Figure 4 that the dimensionless pressures produced using PICIN are very similar to those obtained using the δ-SPH model (cf. Figure 3 of [21]). Moreover, the free surface evolution is also in very close agreement with [21] and exhibits none of the fragmentation of the WCSPH and ISPH results. In Figure 5 the left panel shows a comparison with the experimental data of [6] for the pressure load at a point on the wall that is $73H/75$ above the bed. Also shown in the figure for comparison are the δ-SPH results. As noted in [21], the disparity between the numerical and experimental results that occurs when $t(gH^{-1})^{\frac{1}{2}} > 5.7$ is due to the formation of the

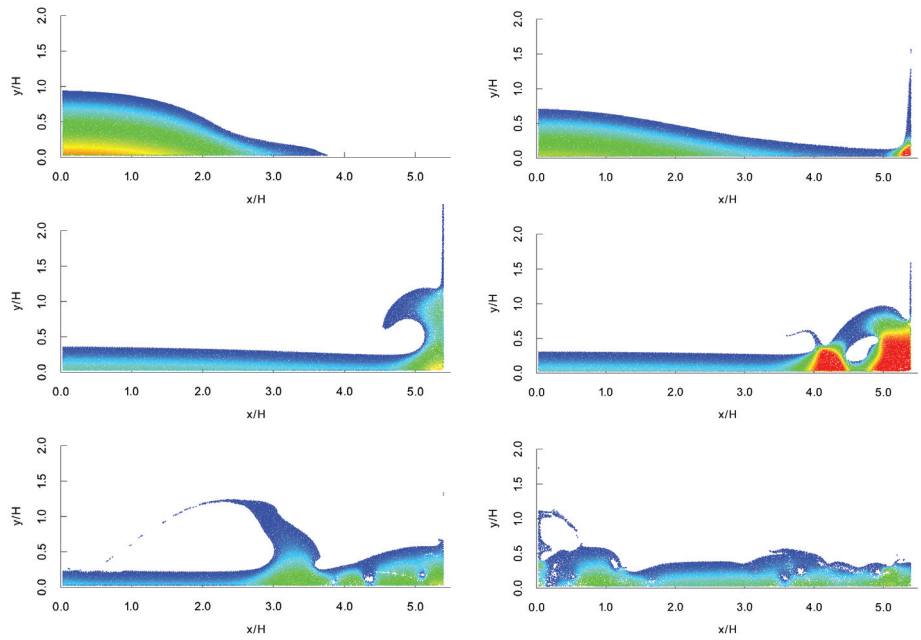

FIG. 4. *Snapshots of dimensionless pressure* $p^* = p(\rho g H)^{-1}$ *interpolated at particle locations for the dam-break Test* 1. *The values of* p^* *range between* 0 *(blue) and* 1 *(red). Snapshots are at dimensionless times* $t^* = t(gH^{-1})^{\frac{1}{2}} = 1.50$, 3.00, 5.70, 6.45, 8.42, *and* 12.00.

internal cavity which requires two-phase modeling in order to simulate the cushioning effect of the air.

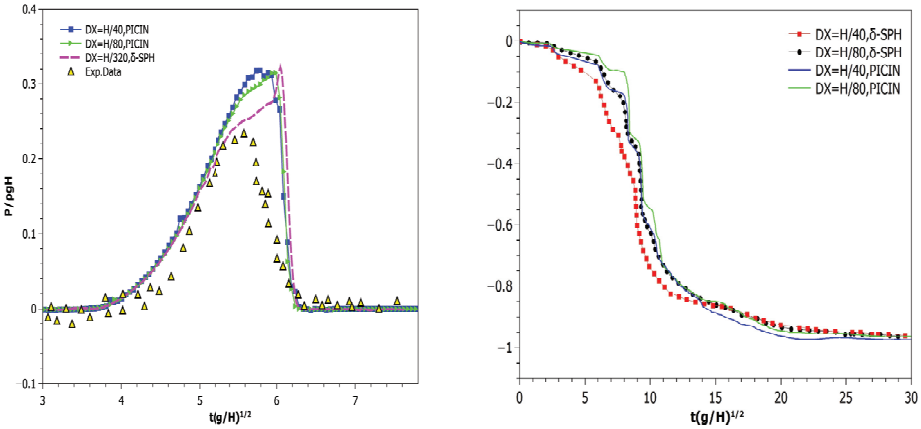

FIG. 5. *Left: Comparison of experimental,* δ*-SPH, and PICIN pressure loads for the dam-break test at a point* $73H/75$ *up the vertical wall. Right: Nondimensional mechanical energy evolution* $E_{mech} = (E_k + E_p - E_p^{(1)})/(E_p^{(1)} - E_p^{(2)})$ *for different levels of spatial resolution. The nondimensionalization is based on the potential energy imbalance following* [21].

Nevertheless, for this demanding test the PICIN results are in good agreement

with the δ-SPH results. The right panel of Figure 5 shows the dimensionless mechanical energy $E_{mech} = (E_k + E_p - E_p^{(1)})/(E_p^{(1)} - E_p^{(2)})$ of the flow computed using both the PICIN model and the δ-SPH model of [21]. Here E_k and E_p denote the total kinetic and potential energies, respectively, of the (closed) system. The mechanical energy is made dimensionless using the difference between the start and finish potential energies, i.e., $E_p^{(1)} - E_p^{(2)}$. It can be seen that the PICIN and δ-SPH results are very similar, with both models exhibiting a large amount of dissipation following the first splash-up. Figure 6 shows the PICIN prediction for the leading edge (tip) of the dam-break wave compared with the experimental data and the WCSPH predictions presented in Figure 5 of [15].

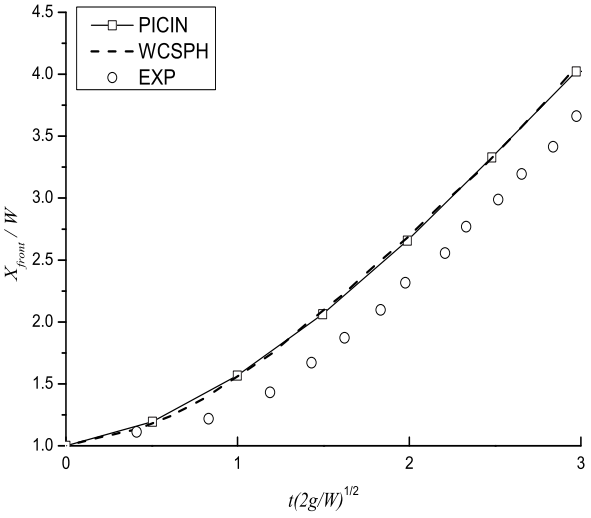

FIG. 6. *Comparison of dam-break tip position between PICIN, weakly compressible SPH (WCSPH), and experimental data (EXP).*

Figure 6 illustrates that the PICIN results compare reasonably well with the experimental data and are very similar to those obtained using a state-of-the-art SPH solver. For this case $W = H/2$ and $D = 2H$. We note that, as with SPH, when using particles alone the exact location of the free surface is hard to identify precisely. PICIN, however, has the advantage that the location of the free surface has been tracked on the mesh using the fluid SDF and can be identified by the $\phi = 0$ iso-contour.

This test used an Intel Core$^{\text{TM}}$i7–4600U (CPU@2.10Ghz $\sim$ 2.70GHz) laptop with 8Gb RAM employing a Windows OS. The CPU time for 6s of simulation time was 400s with $\Delta x = H/40$ and 4680s with $\Delta x = H/80$.

Test 2: Viscous free surface flow around a circular cylinder. This test considers low Reynolds number, high Froude number, open channel flow past a circular cylinder close to a free surface. The same test was recently simulated by [4] using the δ-SPH model of [21]. The problem set-up is given in Figure 7. As the cylinder is fixed, it is considered part of the domain boundary $\Gamma = \Gamma(\boldsymbol{x})$ and treated via the

approach detailed in section 3.2.

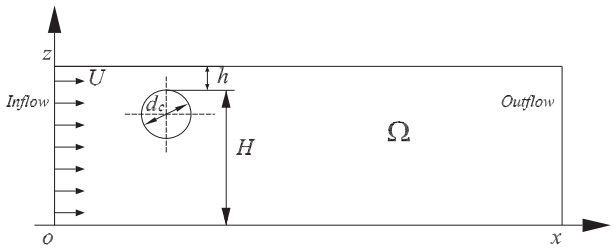

FIG. 7. *Problem set-up for Test* 2.

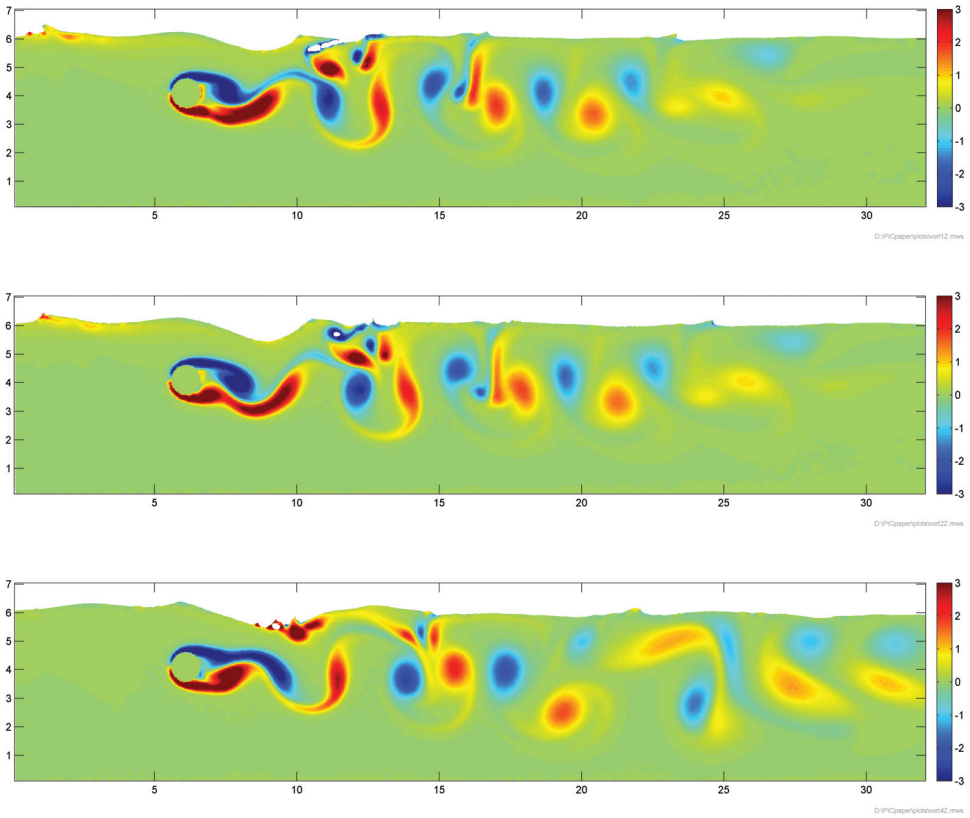

FIG. 8. *Snapshots of dimensionless vorticity* $\nabla \times \boldsymbol{u}(dg^{-1})^{\frac{1}{2}}$ *for Test* 2. *Snapshots are at dimensionless times* $t^* = (gd^{-1})^{\frac{1}{2}}t = 26.64,\ 27.58,\ 37.91$.

Boundary conditions at the upstream and downstream boundary are implemented using a ghost-cell (zeroth-order extrapolation) type approach on the grid. Particles that leave the domain at the downstream boundary are reintroduced at the upstream boundary with a random offset, following a regular distribution within the initial water

depth, and newly prescribed velocity. Following [4], the water depth $\hat{H} = (H+h)$ is set equal to $6d_c$, with d_c being the cylinder diameter. The distance h between the cylinder top and the undisturbed free surface is used to denote the cylinder submergence. Uniform inflow velocity $\boldsymbol{u} = [U, 0]^T$ is imposed on the upstream boundary. Here we present numerical simulations for a case where the cylinder is fully submerged such that ${hd_c}^{-1} = 1.5$ with $d_c = 1.0$. The cylinder is placed at $x = 6d_c = 6m$, and the overall length of the channel is $32d_c = 32m$. To enable comparison with the numerical results presented in [4], we use $Fr = U(g\hat{H})^{-\frac{1}{2}} = 1.0$, where Fr is the characteristic Froude number. The Reynolds number $Re = 200$.

For the PICIN results presented here we used $\Delta x = \Delta z = \hat{H}/60$, giving 322 cells in the x-direction and 75 cells in the z-direction and a total of 76,960 fluid particles. The Courant number was set to 0.5.

Figure 8 shows snapshots of the dimensionless vorticity for three times that illustrate the development of the flow. Figure 9 shows snapshots of the extent of vertical mixing at the same times as in Figure 8. We note that both figures were produced employing a triangulation and an associated linear interpolation of the particle data. The results shown in Figure 8 are very similar to those presented in [4], with complex vorticity interaction between that induced by wave breaking and the cylinder wake. The consequence of this on the intensity of the vertical mixing is illustrated in Figure

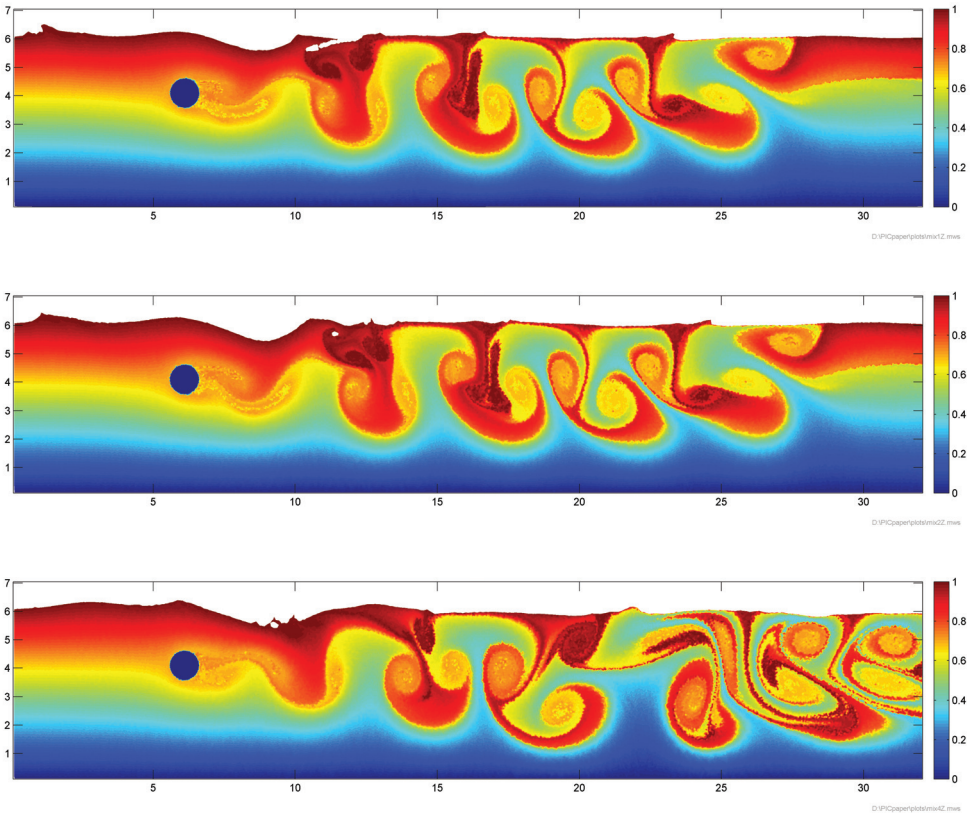

FIG. 9. *Snapshots of vertical mixing extents for Test* 2. *Snapshots are at the dimensionless times given in Figure* 8.

9. We note that the bottom panel of Figure 9 is remarkably similar to Figure 7 of [4], which shows the δ-SPH predictions of vertical mixing at approximately the same time.

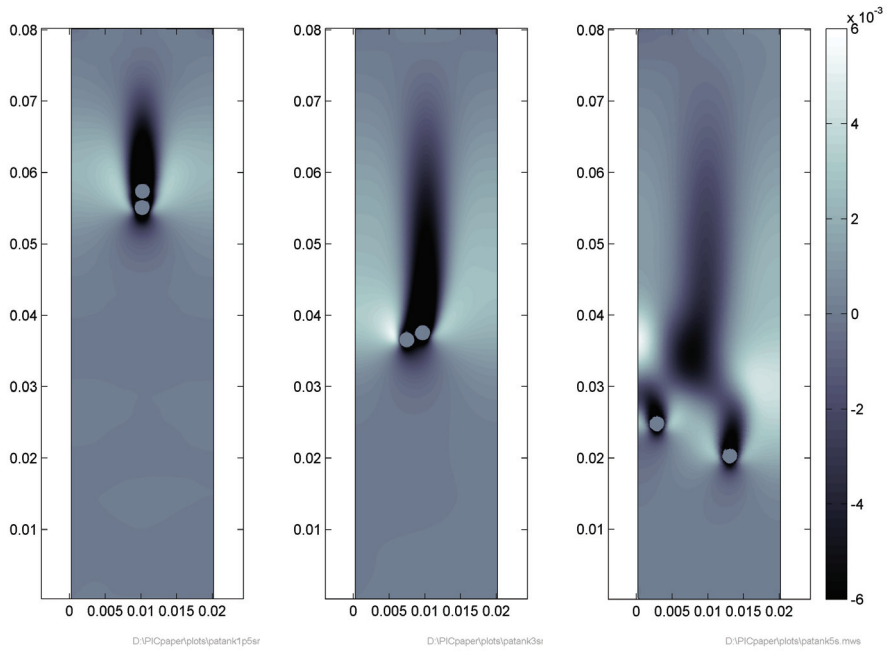

FIG. 10. *Snapshots of vertical velocity contours at $t = 1.5$, 3.0, and 5s for Test 3. The snapshots are chosen to show the three stages of drafting, kissing, and tumbling of the particles.*

The PICIN results also exhibit the cyclical (plunging) wave breaking events downstream of the cylinder observed in the δ-SPH simulation of [4]. The predicted drag coefficient C_D is relatively stable throughout the duration of the run with a mean value of 1.6. This is in reasonable agreement with the mean value of around 1.45 obtained by [4]. PICIN predictions for the lift coefficient C_L oscillations have an amplitude of around 0.5, which is 33% higher than the amplitude of around 0.375 in Figure 3 of [4]. Both C_D and C_L were computed following the methodology given in [3]. For this test the predicted Strouhal number is $St = f_S d_c \boldsymbol{u}_\infty{}^{-1} = 0.23$, where f_S is the vortex shedding frequency and $\boldsymbol{u}_\infty$ is the free stream velocity. We note that the value of St predicted by the PICIN model using $< 80,000$ fluid particles is identical to the value predicted by the δ-SPH model used by [4], the authors of which used 2 million particles.

The CPU time for 12s of simulation time was 68400s with $\Delta x = \Delta z = \hat{H}/60$.

Test 3: Sedimentation of two circular particles. This final test case is chosen to illustrate the ability of PICIN to handle problems that involve full two-way fluid-solid coupling. Following [30] and [29], we consider the sedimentation of two perfectly circular particles in water. The domain boundary $\Gamma = \Gamma(\boldsymbol{x})$ comprises a tank $2cm$ wide (x-direction) and $8cm$ tall (z-direction). To enable comparison with [29] the kinematic viscosity of water is set at $1 \times 10^{-06} m^2/s$, and the water density is $1000 Kgm^{-3}$. The particle density is $1010 Kgm^{-3}$, and the radius is $0.1cm$ for both particles. Gravity acts in the negative z-direction. The simulation is started at $t = 0s$

by dropping the two particles at the center of the channel at a height of 7.2cm and $6.8cm$. For the results presented here $\Delta x = \Delta z = 0.02cm$, and a total of $160,006$ particles were seeded. A fixed time-step of $t = 0.0025s$ was used, and the particle resampling algorithm was employed every 15 time-steps in order to get a smoother fluid particle distribution around the circular particles. Figure 10 shows that the particles follow the expected regime of drafting, kissing, and tumbling.

After tumbling, both particles continue to fall at an approximately constant velocity of just under 1cms^{-1}. Based on this terminal velocity U_c the Reynolds number is approximated, making use of the particle diameter d as $Re = U_c d\nu^{-1} \approx 18$. The results obtained using PICIN are in good agreement with those presented in [29]. Detail of the flow field around the circular particles is shown in Figure 11. Time series of the vertical velocities for both particles are given in Figure 12.

The vertical-velocity time series is quantitatively similar to that presented in [29]; however, it is not identical, with differences appearing once the kissing phase begins. We note, as did the author of [29], that tumbling is the manifestation of an instability and is affected by both the accuracy of the solution procedure and the modeling of the collision. Thus, the predictions of different numerical solvers are highly unlikely to be identical.

The CPU time for 6s of simulation time was 4320s with $\Delta x = \Delta z = 0.02$cm.

5. Conclusion. For incompressible free surface flows the full particle PIC model described here (PICIN) is an attractive alternative to both modern Eulerian schemes, which typically employ a volume of fluid (VOF) or level set free surface treatment, and fully Lagrangian schemes such as MPS and SPH. The PICIN model is able to simulate complex free surface flows in arbitrary domains with both one-way and two-way fluid-structure interaction. The three test cases presented have been chosen to illustrate this capability as well as show the range of problems that the PICIN methodology can successfully simulate. As discussed, the PICIN results are quantitatively in line with the numerical results presented by other researchers for all three test cases shown. We believe that the use of the full particle PIC methodology represents a promising avenue of research for incompressible free surface flows. In particular, the approach seems very promising for flows that involve violent wave impacts with (moving) structures. We stress that we believe the CPU times presented here could be reduced significantly with some optimization of the PICIN source code. Extensive validation of the PICIN solver against experimental data will be presented in a forthcoming companion paper [7].

Appendix A. Computing length fractions for fixed solid boundaries. This appendix explains how the length fractions open to flow (required when using the cut-cell approach to represent static solid boundaries) are approximated in the PICIN model. Referring to Figure 13, the length of the cell open to flow, i.e., the value of $S_{1,\ldots,4}$, can be determined directly from the solid signed distance function (SDF). The solid SDF denoted $\phi_S = \phi_S(\boldsymbol{x}, t)$ is assumed to be known at all cell vertices, allowing the values of $S_{1,\ldots,4}$ to be computed via linear interpolation. An expression for the value of S_1 computed in this manner is

$$S_1 = \begin{cases} \delta \frac{\phi_{s4}}{(\phi_{s4} - \phi_{s1})} & \text{if} \quad \phi_{s1} < 0 \text{ and } \phi_{s4} > 0, \\ \delta \frac{\phi_{s1}}{(\phi_{s1} - \phi_{s4})} & \text{if} \quad \phi_{s1} > 0 \text{ and } \phi_{s4} < 0, \\ 0 & \text{if} \quad \phi_{s1} < 0 \text{ and } \phi_{s4} < 0, \\ \delta & \text{if} \quad \phi_{s1} > 0 \text{ and } \phi_{s4} > 0; \end{cases} \tag{A.1}$$

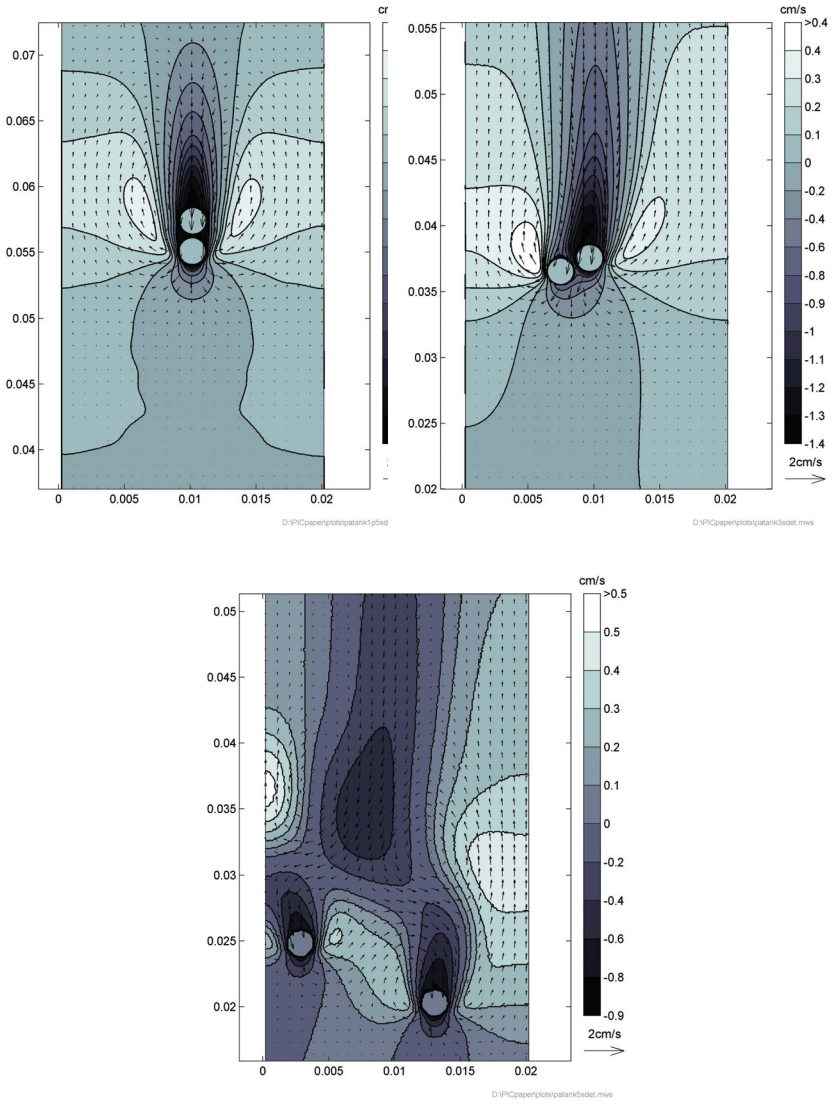

FIG. 11. *Detail of the fluid velocity field in the particle vicinity for Test* 3. *The contours are of vertical velocity, and the snapshots are at the same times as in Figure* 10.

the values of S_2, S_3, and S_4 can be determined in a similar fashion. Note that $\delta = \Delta x$ for S_2 and S_4 and $\delta = \Delta z$ for S_1 and S_3.

Appendix B. Computing the solid fraction θ for nonfixed solid boundaries. This appendix explains how the solid weighting fraction θ (required when using the DLM approach to represent solid boundaries) is approximated in the PICIN model. For cells that are fully occupied by solids or fluids, the coefficient θ is simply taken as 1.0 or 0.0. When a cell is partially cut by the solid boundary, it is assumed that the solid boundary is a straight line and can cut one, two, or three sides of a cell, as shown in Figure 14.

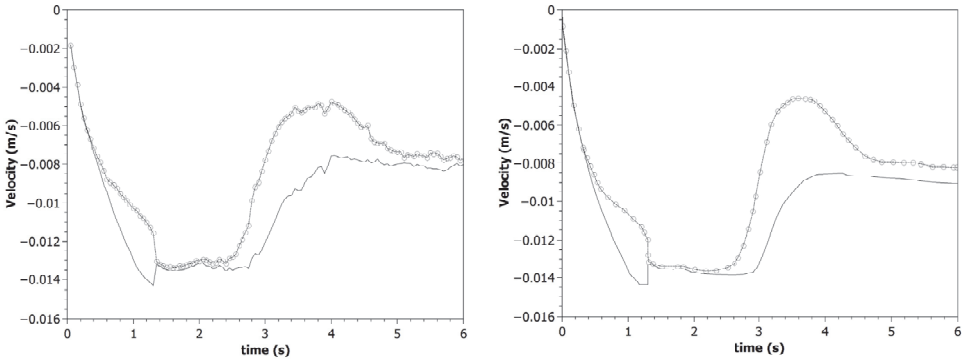

FIG. 12. *Time-series of PICIN predictions (left) and those of* [29] *(right) of vertical velocity for the two circular particles in Test* 3. *The solid line represents the velocity of the lagging particle, and the dashed line with circles represents the velocity of the leading particle.*

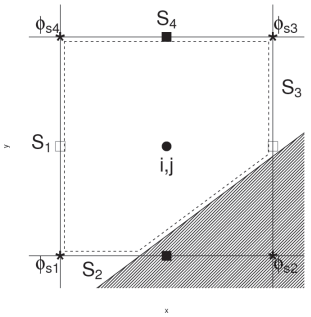

FIG. 13. *A typical solid boundary cut-cell illustrating the length fractions of faces open to flow* $S_{1,\ldots,4}$ *and the location of the solid SDF* $\phi_{s1,\ldots,s4}$ *used to compute them.*

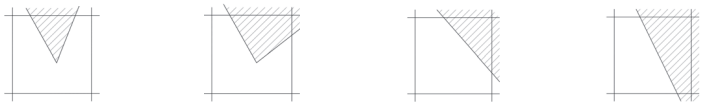

FIG. 14. *Schematic showing various solid configurations in partially solid cells.*

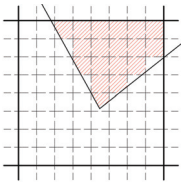

FIG. 15. *Schematic showing the subdivision of a cell partially occupied by a solid. The subcells that contribute to the value of* θ *are shaded.*

The value of θ in cases (c) and (d) can be computed using a simple trapezoidal area rule. Computation of θ in cases (a) and (b) requires more careful consideration.

The technique employed here first divides the partially occupied cell into 64 subcells and then detects whether the subcell center is inside the solid or not. The subcell is considered to be occupied by a solid if its center is inside the solid region, as shown in Figure 15. The final value for θ is taken to be the ratio of the total area of occupied subcells to the area of the full-sized containing cell.

Acknowledgments. The authors are grateful to Dr. A. Dimakopoulos (HR Wallingford) and the two anonymous reviewers, whose suggestions have served to greatly improve the final paper. Use of the MATLAB based Mermaid software developed by Dr. T. Benson (HR Wallingford) to plot model results is also acknowledged.

REFERENCES

[1] D. Adalsteinsson and J. A. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys, 148 (1999), p. 2–22.

[2] R. Ando, N. Thurey, and R. Tsuruno, *Preserving fluid sheets with adaptively sampled anisotropic particles*, IEEE Trans. Visualization Comput. Graphics, 18 (2012), pp. 1202–1214.

[3] E. Balaras, *Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations*, Comput. & Fluids, 33 (2004), pp. 375–404.

[4] B. Bouscasse, A. Colagrossi, S. Marrone, and A. Souto-Iglesias, *Viscous flow past a circular cylinder below a free surface*, in Proceedings of the 33rd ASME International Conference on Ocean, Offshore and Arctic Engineering, San Fransisco, CA, 2014, OMAE2014-24488.

[5] J. U. Brackbill and H. M Ruppel, *FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions*, J. Comput. Phys., 65 (1986), pp. 314–343.

[6] B. Buchner, *Green Water on Ship-Type Offshore Structures*, Ph.D. thesis, Delft University of Technology, Delft, The Netherlands, 2002.

[7] Q. Chen, D. M. Kelly, J. Zang, and A. Dimakopoulos, *PICIN: A Particle-In-Cell Solver for Incompressible Free Surface Flows with Two-Way Fluid-Solid Coupling. Part* 2: *Validation*, in preparation.

[8] A. J. Chorin, *Numerical solution of the Navier–Stokes equations*, Math. Comput., 22 (1968), pp. 745–762.

[9] A. J. Chorin and J. E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, 1979.

[10] A. Colagrossi, B. Bouscasse, M. Antuono, and S. Marrone, *Particle packing algorithm for SPH schemes*, Comput. Phys. Comm., 183 (2012), pp. 1641–1653.

[11] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the Poisson equation on irregular domains*, J. Comput. Phys., 176 (2002), pp. 205–227.

[12] F. H. Harlow, *A Machine Calculation Method for Hydrodynamic Problems*, Technical report LAMS-1956, Los Alamos Scientific Laboratory, Los Alamos, NM, 1955.

[13] F. H. Harlow, *The particle-in-cell computing method for fluid dynamics*, in Methods in Computational Physics, B. Alder, ed., Academic Press, New York, 1964, pp. 319–343.

[14] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[15] J. P. Hughes and D. I. Graham, *Comparison of incompressible and weakly-compressible SPH models for free-surface water flows*, J. Hydraulic Res., 48 (2010), pp. 105–117.

[16] C. B. Jiang, J. Chen, H. S. Tang, and Y. Z. Cheng, *Hydrodynamic processes on a beach: Wave breaking, up-rush and backwash*, Commun. Nonlinear Sci. Numer. Simul., 16 (2011), pp. 3126–3139.

[17] S. Koshizuka, A. Nobe, and Y. Oka, *Numerical analysis of breaking waves using the moving particle semi-implicit method*, Internat. J. Numer. Methods Fluids, 26 (1998), pp. 751–769.

[18] P. Lin and P. L. F. Liu, *A numerical study of breaking waves in the surf zone*, J. Fluid Mech., 359 (1998), pp. 239–264.

[19] S. J. Lind, R. Xu, P. K. Stansby, and B. D. Rogers, *Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves*, J. Comput. Phys., 231 (2012), pp. 1499–1523.

[20] G. Markham and M. V. Proctor, *C.E.G.B. report*, Tech. report TRPD/L/0063/M82, CEGB, London, 1983.
[21] S. Marrone, M. Antuono, A. Colagrossi, G. Colicchio, D. Le Touze, and G. Graziani, *δ-SPH model for simulating violent impact flows*, Comput. Methods Appl. Mech. Engrg., 200 (2011), pp. 1526–1542.
[22] J. J. Monaghan, *SPH without a tensile instability*, J. Comput. Phys., 159 (2000), pp. 290–311.
[23] J. J. Monaghan, *Smoothed particle hydrodynamics and its diverse applications*, Annu. Rev. Fluid Mech., 44 (2012), pp. 323–346.
[24] J. J. Monaghan and J. C. Lattanzio, *A refined method for astrophysical problems*, Astronomy Astrophys., 149 (1985), pp. 135–143.
[25] J. P. Morris, P. J. Fox, and Y. Zhu, *Modeling low Reynolds number incompressible flows using SPH*, J. Comput. Phys., 136 (1997), pp. 214–226.
[26] Y. Ng, C. Min, and F. Gibou, *An efficient fluid-solid coupling algorithm for single-phase flows*, J. Comput. Phys., 228 (2009), pp. 8807–8829.
[27] B. D. Nichols and C. W. Hirt, *Improved free surface boundary conditions for numerical incompressible-flow simulations*, J. Comput. Phys., 8 (1971), pp. 434–448.
[28] W. F. Noh, *CEL: A time-dependent, two-space-dimensional, coupled Eulerian-Lagrange code*, in Methods in Computational Physics, B. Alder, ed., Academic Press, New York, 1964, pp. 117–179.
[29] N. A. Patankar, *A formulation for fast computations of rigid particulate flows*, in Center for Turbulence Research Annual Research Briefs 2001, Stanford University, Stanford, CA, 2001, pp. 185–196.
[30] N. A. Patankar, P. Singh, D. D. Joseph, R. Glowinski, and T.-W. Pan, *A new formulation of the distributed Lagrange multiplier/ fictitious domain method for particulate flows*, Int. J. Multiphase Flow, 26 (2000), pp. 1509–1524.
[31] W. H. Press, S. A. Teulowsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in FORTRAN, The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, UK, 1994.
[32] A. Ralston, *Runge–Kutta methods with minimum error bound*, Math. Comp., 16 (1962), pp. 431–437.
[33] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, 9th ed., Cambridge University Press, Cambridge, UK, 2008.
[34] H. Zhao, *A fast sweeping method for Eikonal equations*, Math. Comp., 74 (2004), pp. 603–627.
[35] Y Zhu and R Bridson, *Animating sand as a fluid*, in Proceedings of ACM SIGGRAPH 2005, ACM, New York, 2005, pp. 965–972.